

ON COMPUTING HIGHER-ORDER ALEXANDER MODULES OF KNOTS

PETER D. HORN

ABSTRACT. Cochran defined the n th-order integral Alexander module of a knot in the three sphere as the first homology group of the knot's $(n+1)$ th-iterated abelian cover. The case $n = 0$ gives the classical Alexander module (and polynomial). After a localization, one can get a finitely presented module over a principal ideal domain, from which one can extract a higher-order Alexander polynomial. We present an algorithm to compute the first-order Alexander module for any knot. Included in this algorithm is a solution to the word problem in finitely presented $\mathbb{Z}[\mathbb{Z}]$ -modules.

1. INTRODUCTION

Given a diagram for a knot K in S^3 , there is a well-known algorithm for producing the Wirtinger presentation of $\pi_1(S^3 \setminus K)$ [Rol76, Section 3.D]. While the knot group is a complete knot invariant (see [Wal78]), using these presentations is an impractical tool for distinguishing knots. Since any knot's exterior has $H_1(S^3 \setminus K) \cong \mathbb{Z}$, abelianizing the fundamental group is not at all a useful invariant for distinguishing knots. A logical next step would be to find an intermediate quotient, $Q(K)$, that is more discerning than H_1 and more tractable than π_1 :

$$\pi_1(S^3 \setminus K) \twoheadrightarrow Q(K) \twoheadrightarrow H_1(S^3 \setminus K)$$

Or better yet, one may hope to find a module $M(K)$ on which the group $Q(K)$ -acts. With this strategy in mind, Cochran defined the higher-order Alexander modules of a knot [Coc04]. The *derived series of a group* G is defined by $G^{(0)} = G$ and $G^{(n+1)} = [G^{(n)}, G^{(n)}]$, where the square brackets denote commutators. Throughout this paper we are interested in knot groups $G = \pi_1(S^3 \setminus K)$. Put simply in algebraic terms, the n th-order integral Alexander module of K is $G^{(n+1)}/G^{(n+2)}$ as a right $\mathbb{Z}[G/G^{(n+1)}]$ -module (the action is conjugation in the group). For $n = 0$, this is the classical Alexander module, which has a topological interpretation as the first homology of the universal abelian cover of the knot complement. The higher-order Alexander modules we consider here are different from those defined before in Cochran-Orr-Teichner [COT03, Sections 2 and 3]; the Cochran-Orr-Teichner modules are defined using a localization stemming from a coefficient system on a 4-manifold in which the knot K is slice. The modules considered here only depend on the knot K .

Let $\mathcal{A}_n^{\mathbb{Z}}(K)$ denote Cochran's n th-order integral Alexander module. In [Coc04], Cochran constructs a localization of the coefficient ring and considers the n th-order (localized) Alexander module $\mathcal{A}_n(K)$. A precise definition will follow in Section 2. As in the classical case, one can extract an integer-valued invariant $\delta_n(K)$, which is the degree of an “ n th-order Alexander polynomial.” These degrees give lower bounds for the *genus of a knot*, which is the minimal genus of all orientable surfaces in S^3 whose boundary is the given knot.

Theorem 1 (Theorems 7.1 and Corollary 9.2 of [Coc04]). *If K is a knot in S^3 with non-trivial Alexander polynomial, then $\delta_0(K) \leq \delta_1(K) + 1 \leq \cdots \delta_n(K) + 1 \leq \cdots \leq 2 \text{ genus}(K)$. Furthermore, given any $n \geq 0$, there exists a knot where $\delta_n(K) < \delta_{n+1}(K)$.*

Thus, the higher δ_n give better lower bounds for the genus of a knot. For the ‘ $<$ ’ part of the theorem, Cochran constructs knots by a satellite operation called ‘infection.’ Many positive results in the so-called ‘higher-order knot theory’ have been proven using satellite operations, but satellite knots are not generic. The higher-order degrees δ_n are the most alluring of all the higher-order knot invariants for

The author was partially supported by NSF DMS-1258630.

two reasons: they are defined for all knots, and they ‘should be’ algorithmically computable. Since the δ_n were defined, several people have made successful computations by hand [LM08], [Hol10]. In a related line of inquiry, Harvey successfully computed degrees of higher-order Alexander polynomials of 3-manifolds [Har05, Sections 6 and 8]. Such calculations are tedious but always seem to work out with enough perseverance.

The purpose of this paper is to describe a practical algorithm to compute $\delta_1(K)$ for any knot K , answering question 13 of [FV11]. Using this algorithm, we compute δ_1 for many low crossing knots. All knots with ten crossings or fewer have $\delta_0 = 2 \text{ genus}(K)$, and so we know δ_1 by Theorem 1. It is easily seen from the definitions in Section 2 that if the classical Alexander polynomial of K is trivial, then all the $\delta_n(K) = 0$. The remaining knots with eleven crossings for which $0 < \delta_0 < 2 \text{ genus}$ are 11_{n45} , 11_{n67} , 11_{n73} , 11_{n97} , and 11_{n152} . We exhibit the utility of the algorithm to compute $\delta_1(11_{n67})$ and $\delta_1(12_{n293})$.

Theorem 2. $\delta_1(11_{n67}) = 3$ and $\delta_1(12_{n293}) = 3$.

The classical Alexander polynomial of 11_{n67} is $2 - 5t + 2t^2$, and its genus is 2. This is the simplest known example of a knot where δ_1 is a better bound on the genus than δ_0 . (The only simpler possible example is 11_{n45}). Since $3 + 1 = 2 \cdot 2$ (as in Theorem 1), we conclude that $\delta_n(11_{n67}) = 3$ for all $n \geq 1$. We remark that the classical Alexander module of 11_{n67} is cyclic; it matches that of the stevedore knot, 6_1 , which has genus one. The calculation of $\delta_1(11_{n67})$ took 11324 seconds on a machine with an Intel i5 2.5 GHz quad-core processor.

The knot 12_{n293} also has genus 2, and its classical Alexander polynomial is $2 - 3t + 2t^2$.

For each of 11_{n67} and 12_{n293} , there exists another knot in the tables with isomorphic knot Floer homology; the computation was verified by Gridlink [Cul]. These other knots are 11_{n97} and 12_{n519} , respectively. We were unable to find Wirtinger presentations for these knots that would allow our program to compute δ_1 in a reasonable amount of time, which leaves the question: does knot Floer homology detect δ_1 ?

In their survey on twisted Alexander polynomials [FV11], Friedl and Vidussi ask whether the higher-order Alexander polynomials detect mutation, and as of the writing of this paper, their question remains open.

2. DEFINITIONS

The definitions of higher-order Alexander modules are due to Cochran [Coc04].

Let K be a knot in S^3 and $G = \pi_1(S^3 \setminus K)$. Let $\Gamma_n = G/G^{(n+1)}$, where $G^{(n+1)}$ is the $(n+1)$ th term of the derived series of G . There is a right action of Γ_n on $G^{(n+1)}/G^{(n+2)}$ by conjugation:

$$g * h = h^{-1} g h, \quad \text{for all } h \in \Gamma_n, g \in G^{(n+1)}/G^{(n+2)}$$

Let M_n denote the $(n+1)$ th iterated (maximal) abelian cover of $S^3 \setminus K$, so that $\pi_1(M_n) \cong G^{(n+1)}$. Then $H_1(M_n; \mathbb{Z}) \cong G^{(n+1)}/G^{(n+2)}$, and this homology group has a right action by Γ_n (viewed as either deck translations or conjugation).

Definition 1. The n th-order integral Alexander module of K , denoted $\mathcal{A}_n^{\mathbb{Z}}(K)$, is the right $\mathbb{Z}\Gamma_n$ -module

$$\mathcal{A}_n^{\mathbb{Z}}(K) := G^{(n+1)}/G^{(n+2)} \cong H_1(S^3 \setminus K; \mathbb{Z}\Gamma_n) \cong H_1(M_n; \mathbb{Z})$$

The group Γ_n is *poly-(torsion-free) abelian* (PTFA), i.e. it admits a series of normal subgroups $\{e\} \triangleleft A_k \triangleleft A_{k-1} \triangleleft \cdots \triangleleft A_0 = \Gamma_n$ so that each A_i/A_{i+1} is torsion-free abelian. To see this, take $A_i = G^{(i)}/G^{(n+1)}$ and recall that $A_i/A_{i+1} = G^{(i)}/G^{(i+1)}$ is torsion-free abelian [Str74]. We will use the fact that $\mathbb{Z}\Gamma_n$ imbeds in its classical quotient field (in general $\mathbb{Z}\Gamma_n$ is a noncommutative ring, so its quotient field is a skew field):

Proposition 1 (Proposition 3.2 of [Coc04], see also the Proposition of [Lew72]). *If Γ is PTFA, then $\mathbb{Q}\Gamma$ (and hence $\mathbb{Z}\Gamma$) is a right Ore domain, i.e. $\mathbb{Q}\Gamma$ imbeds in its classical right ring of quotients \mathcal{K} , which is a skew field.*

Let \mathcal{K}_n denote the quotient field of $\mathbb{Z}\Gamma_n$. There is an important intermediate ring R_n ($\mathbb{Q}\Gamma_n \subset R_n \subset \mathcal{K}_n$) which will be convenient for our use. For $n \geq 1$, the kernel of $\pi : G/G^{(n)} \rightarrow G/G^{(1)} \cong \mathbb{Z}$ is just $G^{(1)}/G^{(n)}$. After choosing a splitting $1 \mapsto \mu$ of π , we get an isomorphism

$$\Gamma_n = G/G^{(n+1)} \cong G^{(1)}/G^{(n+1)} \rtimes \mathbb{Z}$$

Any element of Γ_n has a unique expression as $\mu^i g$, where $g \in G^{(1)}/G^{(n+1)}$. There is also a unique expression by writing powers of μ on the right: $\mu^i g = \tilde{g} \mu^i$, where the \sim denotes the action on g by conjugating i times by μ . Thus $\mathbb{Q}\Gamma_n$ is canonically isomorphic (after choosing the splitting) to the skew Laurent polynomial ring $(\mathbb{Q}[G^{(1)}/G^{(n+1)}])[t^{\pm 1}]$. In this skew polynomial ring, the coefficients are not commutative (unless $n = 1$), and the coefficients do not commute with the variable (due to the semidirect product structure).

Let \mathbb{K}_n denote the classical right (skew) field of quotients for $\mathbb{Q}[G^{(1)}/G^{(n+1)}]$, and let $R_n = \mathbb{Z}\Gamma_n (\mathbb{Z}[G^{(1)}/G^{(n+1)}] \setminus \{0\})^{-1}$. After choosing a splitting $1 \mapsto \mu$, there is a canonical isomorphism $R_n \xrightarrow{\cong} \mathbb{K}_n[t^{\pm 1}]$. Since R_n is a localization of $\mathbb{Z}\Gamma_n$ wherein some but not all elements are inverted, we have that $\mathbb{Z}\Gamma_n \subset R_n \subset \mathcal{K}_n$.

Definition 2. The n th-order (localized) Alexander module of K is $\mathcal{A}_n(K) = H_1(S^3 \setminus K; R_n)$.

We now summarize several useful facts about R_n and \mathcal{K}_n :

Theorem 3 (Section 4 of [Coc04]). Suppose we have chosen a meridian (i.e. splitting) $1 \mapsto \mu$ so that $R_n \cong \mathbb{K}_n[t^{\pm 1}]$. Then

- (1) R_n is a right (and left) PID,
- (2) $\mathbb{K}_n[t^{\pm 1}]$ has a well-defined degree function and a Euclidean algorithm,
- (3) $\mathcal{A}_n(K)$ is a finitely-generated torsion module over R_n ,
- (4) R_n is a flat left $\mathbb{Z}\Gamma_n$ -module, so that $\mathcal{A}_n(K) \cong \mathcal{A}_n^{\mathbb{Z}}(K) \otimes_{\mathbb{Z}\Gamma_n} R_n$,
- (5) \mathcal{K}_n is a flat R_n module so $\mathcal{A}_n(K) \otimes_{R_n} \mathcal{K}_n = H_1(S^3 \setminus K; \mathcal{K}_n)$

The structure theorem for finitely-generated modules over commutative PIDs generalizes to the non-commutative case [Coh85]: if M is a finitely generated torsion right R module, where R is a PID, then $M \cong R/e_1 R \oplus \cdots \oplus R/e_k R$ where e_i is a total divisor of e_{i+1} (moreover, this determines the e_i up to similarity).

If one views the e_i as elements of $\mathbb{K}_n[t^{\pm 1}]$, one could define the higher-order Alexander polynomials of K as the product of the e_i . However, this is not a well-defined invariant of K , as the isomorphism $R_n \cong \mathbb{K}_n[t^{\pm 1}]$ depends on a choice of meridian. However, the degree of this polynomial is a knot invariant.

Definition 3 (Definition 5.3 of [Coc04]). For any knot K and any $n \geq 0$, the degree of the n th order Alexander polynomial, denoted $\delta_n(K)$, has several equivalent definitions:

- (1) the sum of the degrees of the $e_i \in R_n \cong \mathbb{K}_n[t^{\pm 1}]$,
- (2) the rank of $\mathcal{A}_n(K)$ as a module over \mathbb{K}_n ,
- (3) the rank of $G^{(n+1)}/G^{(n+2)} \otimes_{\mathbb{Z}\Gamma_n} R_n$ as a module over $\mathbb{Z}[G^{(1)}/G^{(n+1)}]$, or
- (4) the rank of $G^{(n+1)}/G^{(n+2)}$ as a module over $\mathbb{Z}[G^{(1)}/G^{(n+1)}]$

Example 1. If $n = 0$, $\Gamma_0 = G/G^{(1)} \cong \mathbb{Z}$. In this case $\mathbb{K}_0 = \mathbb{Q}$, and $R_0 \cong \mathbb{Q}[t^{\pm 1}]$. Then $\mathcal{A}_0^{\mathbb{Z}}(K)$ is the classical Alexander module $G^{(1)}/G^{(2)}$ over $\mathbb{Z}[t^{\pm 1}]$, and $\mathcal{A}_0(K)$ is the rational Alexander module $\mathcal{A}_0(K) = G^{(1)}/G^{(2)} \otimes_{\mathbb{Z}[t^{\pm 1}]} \mathbb{Q}[t^{\pm 1}]$.

Example 2. Consider the case $n = 1$, which is the main focus of this paper. The group $\Gamma_1 = G/G^{(2)}$ fits into an exact sequence

$$0 \rightarrow G^{(1)}/G^{(2)} \rightarrow \Gamma_1 \rightarrow \mathbb{Z} \rightarrow 0$$

After choosing a preferred meridian, which will be identified with t , we see that any element of Γ_1 can be written uniquely as $t^i g$, where $g \in G^{(1)}/G^{(2)}$. In this case, \mathbb{K}_1 is the quotient field of $\mathbb{Z}[G^{(1)}/G^{(2)}]$,

which is a commutative field! Addition and multiplication in this field work – at least on a symbolic level – as they do in \mathbb{Q} , but one must be especially careful not to divide by zero. Section 4 addresses this concern. While \mathbb{K}_1 is commutative, the coefficients do not commute with the variable in $R_1 \cong \mathbb{K}_1[t^{\pm 1}]$. For example, $tg = \mu g \mu^{-1}t$ in the polynomial ring.

2.1. Outline of the algorithm to compute δ_1 . To compute $\delta_1(K)$, we will take a presentation matrix M for $\mathcal{A}_1(K)$ as a module over $\mathbb{K}_1[t^{\pm 1}]$ and make it upper-triangular. The sum of the degrees of the diagonal entries is $\delta_1(K)$. Note that we do not need the divisibility condition on the diagonal entries as in the structure theorem (cf. Lemma 1). The triangularization process requires only row operations (and some column swaps), which can be achieved by left-multiplication by elementary matrices over $\mathbb{K}_1[t^{\pm 1}]$. The algorithm to upper-triangularize a matrix over $\mathbb{K}_1[t^{\pm 1}]$ is the same as over $\mathbb{Q}[t^{\pm 1}]$. First, multiply each row of M by a suitable power of t to make all entries *honest* polynomials. Pick the smallest degree entry and move it to the top-left entry by row and column swaps. Use row operations to clear the first column; it may be necessary to perform a row swap if the degree of an entry in the $k, 1$ -position (where $k > 1$) falls below the degree of the $1, 1$ -entry. By doing these row operations, the matrix will eventually have a non-zero entry in $1, 1$ and all zeros below that; this follows from items 2 and 3 of Theorem 3. The algorithm continues by moving on to the submatrix gotten by removing the first row and column.

The difficulty with implementing this algorithm lies in the coefficient field \mathbb{K}_1 , which consists of formal quotients of formal \mathbb{Z} -linear combinations of elements in the classical Alexander module, $G^{(1)}/G^{(2)}$, the group operation for which we will denote additively in this paragraph. While this is a commutative field, arithmetic is lengthy. For example, the sum

$$\frac{ma + nb}{oc} + \frac{pd}{qe + rf} = \frac{mq(a + e) + mr(a + f) + nq(b + e) + nr(n + f) + op(c + d)}{oq(c + e) + or(e + f)}$$

seems to have 5 terms in its numerator and two terms in its denominator. But do any of the terms cancel? Is the numerator zero? Logically, the ‘collapsing problem’ in the group ring $\mathbb{Z}H$ is equivalent to the word problem in the group H . However in practice, one would prefer to have a ‘normal form’ for group elements in H to speed up the ring operations in $\mathbb{Z}H$. This will be discussed further in Section 4.

While \mathbb{K}_1 is a commutative field, its arithmetic seems to push the boundaries of the abilities of modern computers. It seems to the author that an implementation to compute δ_n for $n \geq 2$ is hopeless with the current technology, since the fields \mathbb{K}_n with $n \geq 2$ are noncommutative.

3. PREVIOUS AD HOC COMPUTATIONS

For knots with very few crossings (five or fewer), one can calculate δ_1 by hand. Such a calculation is straightforward for two reasons: the size of the matrix you must work with is manageable, and the coefficient field \mathbb{K}_1 is simple, due to the simple structure of \mathcal{A}_0 for low crossing knots. We present two examples to illustrate both the methods involved in the general algorithm and why previous δ_1 computations have failed to give a general algorithm. Compare the strategy in the next example with [Hol10, Example 4.2].

Example 3. Let T denote the trefoil knot. Using the standard torus knot diagram, one can calculate the Wirtinger presentation

$$G := \pi_1(S^3 \setminus T) \cong \langle x_1, x_2, x_3 \mid x_1 x_3 x_1^{-1} x_2^{-1}, x_1 x_3 x_2^{-1} x_3^{-1} \rangle$$

For reasons that will soon become clear, change the generating set to $a := x_1$, $y_1 := x_2 x_1^{-1}$, $y_2 := x_3 x_1^{-1}$. Note that under the abelianization map, a maps to a generator of \mathbb{Z} while y_1 and y_2 map to the identity. We now have the presentation

$$\pi_1(S^3 \setminus T) \cong \langle a, y_1, y_2 \mid ay_2 a^{-1} y_1^{-1}, ay_2 y_1^{-1} a^{-1} y_2^{-1} \rangle$$

We construct the matrix of Fox derivatives [Fox53] whose i, j -entry is the derivative of the i th relation with respect to the j th generator; the Fox calculus is defined by

- $\frac{\partial g_j}{\partial g_i} = \delta_{ij}$, for each i, j

- $\frac{\partial 1}{\partial g_i} = 0$, for each i
- $\frac{\partial uv}{\partial g_i} = \frac{\partial u}{\partial g_i} + u \frac{\partial v}{\partial g_i}$, for each i and for each u, v in the group

where the g_i are the generators of the group. It follows that $\frac{\partial u^{-1}}{\partial g_i} = -u^{-1} \frac{\partial u}{\partial g_i}$ for all u in the group. The Fox matrix for the second presentation is

$$\begin{pmatrix} 1 - ay_2a^{-1} & -1 & a \\ 1 - ay_2y_1^{-1}a^{-1} & -ay_2y_1^{-1} & a - 1 \end{pmatrix}$$

The entries of this matrix lie in the group ring $\mathbb{Z}\pi_1$. Upon writing the group elements in any quotient group G , one obtains a presentation matrix for $H_1(S^3 \setminus T, x_0; \mathbb{Z}G)$, the first homology relative to a basepoint of the covering space whose group of covering transformations is G .

Let us write the Fox matrix ‘over \mathbb{Z} ’ by abelianizing the group elements:

$$\begin{pmatrix} 0 & -1 & t \\ 0 & -t & t - 1 \end{pmatrix}$$

The first column has all zeros by the choice of generators a, y_1, y_2 . Recall that the Fox matrix is a presentation matrix for the first homology of a covering space relative to a basepoint; this, in effect, adds a free generator to $H_1(S^3 \setminus T; \mathbb{Z}[t^{\pm 1}])$, since in the long exact sequence for the pair $(S^3 \setminus T, x_0)$, we have

$$\begin{array}{ccccccc} 0 & \longrightarrow & H_1(S^3 \setminus T; \mathbb{Z}[t^{\pm 1}]) & \longrightarrow & H_1(S^3 \setminus T, x_0; \mathbb{Z}[t^{\pm 1}]) & \longrightarrow & H_0(x_0; \mathbb{Z}[t^{\pm 1}]) \\ & & \downarrow \cong & & \downarrow \cong & & \downarrow \cong \\ & & \text{torsion} & & \text{free} \oplus \text{torsion} & & \mathbb{Z}[t^{\pm 1}] \end{array}$$

By exactness, the free part of $H_1(S^3 \setminus T, x_0; \mathbb{Z}[t^{\pm 1}])$ has rank one.

The free generator in this case is the one corresponding to the first column, and upon eliminating that column, we end up with a presentation for $H_1(S^3 \setminus T; \mathbb{Z}[t^{\pm 1}])$, i.e. the classical Alexander module of T :

$$\begin{pmatrix} -1 & t \\ -t & t - 1 \end{pmatrix}$$

By the definition of y_1, y_2 , y_1, y_2 may be viewed as generators of the classical Alexander module (they lie in the commutator subgroup), and the first and second columns of the above matrix correspond to these generators. In other words, $\mathcal{A}_0^{\mathbb{Z}}(T)$ is generated as a $\mathbb{Z}[t^{\pm 1}]$ -module by y_1 and y_2 with the relations $-y_1 + t * y_2 = 0$ and $-t * y_1 + (t - 1) * y_2 = 0$. One can actually ‘upper-triangularize’ this presentation matrix (using only row operations, which amounts to adding multiples of the relations together) to:

$$\begin{pmatrix} 1 & -t \\ 0 & t^2 - t + 1 \end{pmatrix}$$

Thus $y_1 = t * y_2$ and $(t^2 - t + 1) * y_2 = 0$ in the Alexander module $\mathcal{A}_0^{\mathbb{Z}}(T)$. This description of $\mathcal{A}_0^{\mathbb{Z}}(T)$ will allow us to understand the coefficients in \mathbb{K}_1 .

To compute $\delta_1(T)$, we need a presentation matrix for $\mathcal{A}_1(T) = H_1(S^3 \setminus T; \mathcal{R}_1)$. Since $\mathbb{Z}\Gamma_1 \subset \mathcal{R}_1$, and Γ_1 is a quotient of π_1 , such a presentation matrix can be derived from the Fox matrix above. First, reduce the elements in the Fox matrix according to the quotient map $\pi_1(S^3 \setminus T) \twoheadrightarrow G/G^{(2)} = \Gamma_1$. Recall from Section 2 that any element in Γ_1 has a unique expression (using the semidirect product) once we have chosen a meridian. We have already chosen a distinguished meridian: a . Let us write the Fox matrix with entries written in $G/G^{(2)} \cong G^{(1)}/G^{(2)} \rtimes \mathbb{Z}$, with t generating \mathbb{Z} (we use a when it conjugates

an element of $G^{(1)}/G^{(2)}$, and t when there is no conjugation):

$$\begin{pmatrix} 1 - ay_2a^{-1} & -1 & t \\ 1 - ay_2a^{-1}ay_1^{-1}a^{-1} & -ty_2y_1^{-1} & t-1 \end{pmatrix}$$

Again, this matrix presents $H_1(S^3 \setminus T, x_0; \mathbb{Z}\Gamma_1)$. Since R_1 is a flat $\mathbb{Z}\Gamma_1$ module, so this matrix also presents $H_1(S^3 \setminus T, x_0; R_1)$ since $\mathbb{Z}\Gamma_1 \hookrightarrow R_1$. To get a presentation matrix for $H_1(S^3 \setminus T; R_1)$, we must eliminate one column as we did above for $\mathbb{Z}[t^{\pm 1}]$. Let r_i denote the i th relation and x_j the j th generator in the presentation for G . By the fundamental formula for Fox derivatives [Fox53, eq 2.3],

$$(1) \quad \frac{\partial r_i}{\partial x_j}(x_j - 1) = \sum_{j' \neq j} \frac{\partial r_i}{\partial x_{j'}}(x_{j'} - 1)$$

This implies that the j th column of the Fox matrix is a R_1 -linear combination of the other columns as long as $x_j \in G^{(1)}$ and $x_j \neq 1 \in \mathbb{Z}\Gamma_1$. As long as we can find a generator $y_i \in G^{(1)}$ which is not in $G^{(2)}$, we can (perform column operations according to Equation 1 until the column corresponding to y_i has all zeros and) delete the column corresponding to y_i . This results in a presentation matrix for $H_1(S^3 \setminus T; R_1)$, since $\text{rank}_{R_1} H_1(S^3 \setminus T, x_0; R_1) = 1$, which we will justify now. The existence of the column of zeroes establishes that $\text{rank}_{R_1} H_1(S^3 \setminus T, x_0; R_1) > 0$. One may choose a splitting $H_1(S^3 \setminus T, x_0; R_1) \cong R_1^d \oplus \text{torsion}$ and consider the map $(R_1)^d \rightarrow H_0(x_0; R_1) \cong R_1$ induced by the right most map in the following portion of the long exact sequence for the pair $(S^3 \setminus T, x_0)$:

$$\begin{array}{ccccccc} 0 & \longrightarrow & H_1(S^3 \setminus T; R_1) & \longrightarrow & H_1(S^3 \setminus T, x_0; R_1) & \longrightarrow & H_0(x_0; R_1) \\ & & \downarrow \cong & & \downarrow \cong & & \downarrow \cong \\ & & \text{torsion} & & \text{free} \oplus \text{torsion} & & R_1 \end{array}$$

The map $(R_1)^d \rightarrow R_1$ is injective by the exactness of the sequence above. That $d = 1$ now follows from the rank-nullity theorem over a PID.

The upper-left triangular presentation matrix for $\mathcal{A}_0^{\mathbb{Z}}(T)$ guarantees that $y_2 \neq 1 \in G^{(1)}/G^{(2)}$. Thus, we remove the last column of the Fox matrix over $\mathbb{Z}\Gamma_1$ (which corresponds to y_2) to obtain a square matrix, which we will call the *metabelian Fox matrix*:

$$(2) \quad \begin{pmatrix} 1 - ay_2a^{-1} & -1 \\ 1 - ay_2a^{-1}ay_1^{-1}a^{-1} & -ty_2y_1^{-1} \end{pmatrix}$$

The reader may have noticed that the second column may also be deleted (the element y_1 is nontrivial in $G^{(1)}/G^{(2)}$). We chose the last column because fewer row operations are required to upper-triangularize matrix 2.

Since $\mathbb{Z}\Gamma_1 \subset \mathcal{R}_1 \cong \mathbb{K}_1[t^{\pm 1}]$, matrix 2 is a presentation matrix for $\mathcal{A}_1(T)$, as discussed above. We choose in our algorithm to use only row operations at this point because they are achieved by multiplications on the left by elementary matrices over $\mathbb{K}_1[t^{\pm 1}]$. Column operations are equally valid but require the extra mental effort of remembering to do multiplication *on the right* in the noncommutative ring $\mathbb{K}_1[t^{\pm 1}]$!

We add $-ty_2y_1^{-1}$ times the first row to the second and switch columns:

$$\begin{pmatrix} -1 & 1 - ay_2a^{-1} \\ 0 & t(ay_2a^{-1}y_2y_1^{-1} - y_2y_1^{-1}) + 1 - ay_2a^{-1}ay_1^{-1}a^{-1} \end{pmatrix}$$

At this point, adding the degrees of the diagonal polynomials gives $\delta_1(T)$ by Lemma 1, but we must be careful. The -1 has degree zero. Linear-looking polynomials such as the bottom-right entry do not necessarily have degree one; for example, $tx + y$ will have degree 1 if and only if $y \neq 0$ and $x \neq 0 \in \mathbb{K}_1$. Recall that \mathbb{K}_1 is the quotient field of $\mathbb{Z}[G^{(1)}/G^{(2)}]$. We can determine whether the coefficients are zero using the presentation matrix for $\mathcal{A}_0^{\mathbb{Z}}(T)$. The t -coefficient $ay_2a^{-1}y_2y_1^{-1} - y_2y_1^{-1}$ is zero if and only if

$ay_2a^{-1}ay_1^{-1}a^{-1} = 1$ in $G^{(1)}/G^{(2)}$. This equality in the group can be translated into an equality in the $\mathbb{Z}[t^{\pm 1}]$ -module $\mathcal{A}_0^{\mathbb{Z}}(T)$: $t * y_2 - t * y_1 = 0$? Since $y_1 \neq y_2$ in $\mathcal{A}_0^{\mathbb{Z}}(T)$, the t -coefficient in the linear-looking polynomial is nonzero. We leave it to the reader to verify that the constant term is also nonzero.

We compute $\delta_1(T) = \deg(-1) + \deg(t (ay_2a^{-1}y_2y_1^{-1} - y_2y_1^{-1}) + 1 - ay_2a^{-1}ay_1^{-1}a^{-1}) = 0 + 1 = 1$.

Few choices were made in the computation of $\delta_1(T)$ (choice of preferred meridian, which column to delete). These choices did not affect the end result, and one can easily program a computer how to make these choices. The burden of the computation lies in putting the metabelian Fox matrix in upper-triangular form. In general, many row (and perhaps column operations) are needed, and the \mathbb{K}_1 -coefficients quickly become unruly. While Example 3 was fairly simple, an outline for the general algorithm to compute δ_1 can be gleaned:

Algorithm 1.

- (1) Input: a diagram for a knot K .
- (2) Compute a Wirtinger presentation for $\pi_1(S^3 \setminus K)$ and change the generators.
- (3) Compute the general Fox matrix and abelian Fox matrix.
- (4) Be able to recognize 0 in the ring $\mathbb{Z}\Gamma_1$ so row operations over $\mathbb{K}_1[t^{\pm 1}]$ are possible.
- (5) Decide which column(s) of the Fox matrix can be deleted, and delete one of them to arrive at the metabelian Fox Matrix.
- (6) Put the metabelian Fox matrix in upper-triangular form using valid row operations over $\mathbb{K}_1[t^{\pm 1}]$.
- (7) Compute and add the degrees of the polynomials on the diagonal, giving $\delta_1(K)$.

Proof. We will describe an algorithm for each step and a lemma pertaining to steps 6 and 7. Recall that Definition 3 item 1 requires a *diagonal* matrix that presents $\mathcal{A}_1(K)$ wherein the diagonal entries are total divisors of the subsequent diagonals. The divisibility criterion is immaterial in the computation of the degrees of the diagonal polynomials. Lemma 1 guarantees that an upper-triangular, rather than diagonal, form of the presentation matrix suffices, which will save considerable time by avoiding many tedious row and column operations.

Steps 1, 2 and 3 are straightforward. We use in our implementation a script written by an REU group at Columbia University that converts a knot diagram (PD Code) to a Wirtinger presentation [REU11]. Step 4 is the most difficult. One needs to understand $\mathbb{Z}[G^{(1)}/G^{(2)}]$ well enough to know when an element is zero or nonzero; outside of that, addition, multiplication and inversion in \mathbb{K}_1 behave like the same operations in \mathbb{Q} . The proofs of Theorem 4 given in Sections 4.1, 4.2, and 4.3 settle step 4. Step 5 is then simple as in Example 3: the column corresponding to one of the generators $y_i \in G^{(1)}$ may be deleted if and only if $y_i \neq 0$ in $G^{(1)}/G^{(2)}$. Step 6 is settled by an adaptation to $\mathbb{K}_1[t^{\pm 1}]$ of the algorithm to compute the canonical form of a module over a PID (cf. Section 2.1), although for the purpose of computing δ_1 , one only needs upper triangular and does not need the divisibility criterion as discussed above. Finally, step 7 is straightforward (we use the degree function for Laurent polynomials, so $\deg(t + t^{-2}) = 3$ and $\deg(t^2 + t) = 1$).

□

Several groups of undergraduates have attempted to implement an algorithm to compute δ_1 , and the sticking point tends to be step 4. A strategy that works for many low crossing knots is to compute the abelian Fox matrix, which is a presentation for $\mathcal{A}_0^{\mathbb{Z}}(K) \cong G^{(1)}/G^{(2)}$, and *put it in upper triangular form*. This works for the trefoil in Example 3. The problem is that ‘upper-triangularizing the abelian Fox matrix’ is impossible in general, as we see now.

Example 4. Let K denote the knot in Figure 1. One may compute via Fox calculus or from a Seifert surface a presentation for $\mathcal{A}_0^{\mathbb{Z}}(K)$ to be

$$P = \begin{pmatrix} 3 - 3t & 2 - t \\ 1 - 2t & 3 - 3t \end{pmatrix}$$

The Alexander polynomial of K is $7t^2 - 13t + 7$, an irreducible polynomial over $\mathbb{Z}[t^{\pm 1}]$.

P can be put in upper-triangular form over $\mathbb{Q}[t^{\pm 1}]$, which is a PID, but this is impossible over $\mathbb{Z}[t^{\pm 1}]$, for suppose

$$P \sim \begin{pmatrix} a & b \\ 0 & c \end{pmatrix}$$

where \sim denotes a sequence of invertible row and column operations over $\mathbb{Z}[t^{\pm 1}]$. We may assume without loss of generality that $a = 1$, since $7t^2 - 13t + 7$ is irreducible.

Recall that the first elementary ideal of a module with an $r \times r$ presentation matrix is the ideal \mathcal{E}_1 generated by all $r-1 \times r-1$ minors of the matrix. For a 2×2 matrix such as P , \mathcal{E}_1 is the ideal generated by all of its entries. This ideal is invariant under the operations \sim . For our P , $\mathcal{E}_1 = \langle 2-t, 1-2t \rangle$, since $3-3t$ is the sum of the two off-diagonal entries. If P were upper-triangularizable as above, then $\mathcal{E}_1 = \mathbb{Z}[t^{\pm 1}]$. Then we could write some power of t as $t^m = p(t)(t-2) + q(t)(2t-1)$ for some $p(t), q(t) \in \mathbb{Z}[t]$. Plugging in $t = 2$, we see that $2^m = q(2) * 3$, but 2^m is not divisible by 3, a contradiction.

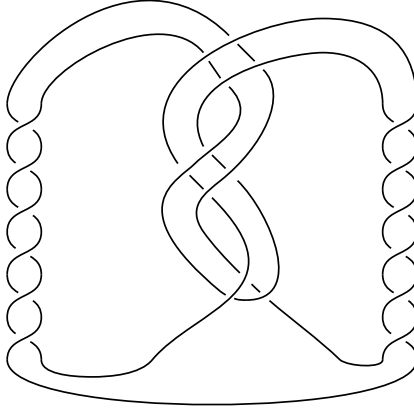


FIGURE 1. A knot with non-upper-triangularizable abelian Fox matrix

We now present the lemma which is crucial in the proof of Algorithm 1.

Lemma 1. *Suppose P is a square presentation matrix for $\mathcal{A}_n(K)$, or more generally for any torsion $\mathbb{K}_n[t^{\pm 1}]$ -module. If P is upper-triangular with diagonal entries d_1, \dots, d_k , then*

$$\sum_{i=1}^k \deg d_i = \text{rank}_{\mathbb{K}_n} \mathcal{A}_n(K) = \delta_n(K)$$

Proof. The claim follows essentially from the Euclidean algorithm. Row (respectively, column) operations are realized by left (respectively, right) multiplication by elementary matrices. Thus, to reduce an entry c in the matrix by another entry b of smaller degree using a row operation, we need to use the ‘left quotient’ $c = qb + r$. Right quotients will be used for column operations. Since $\mathbb{K}_n[t^{\pm 1}]$ is a PID, one can transform P to a diagonal matrix using these operations. Furthermore, it suffices to prove the lemma for 2×2 matrices. For, suppose the lemma has been established for 2×2 matrices. Given

$$P = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix}$$

we may change P to

$$P \rightsquigarrow \begin{pmatrix} a' & 0 & c \\ 0 & d' & e' \\ 0 & 0 & f \end{pmatrix}$$

by only disturbing the first two rows and columns. This may in turn be changed to

$$\begin{pmatrix} a'' & 0 & 0 \\ 0 & d' & e' \\ 0 & 0 & f' \end{pmatrix}$$

by extending the operations on the submatrix

$$\begin{pmatrix} a' & c \\ 0 & f \end{pmatrix}$$

to the entire 3×3 matrix. One continues until P has been completely diagonalized. One continues by induction to prove the lemma for matrices of arbitrary size.

It remains to establish the 2×2 case. Suppose

$$P = \begin{pmatrix} a & b \\ 0 & c \end{pmatrix}$$

Given an element $x \in \mathbb{K}_n[t^{\pm 1}]$, we denote its degree by using uppercase symbols, i.e. $X = \deg x$. Returning our focus to P , we may perform some row and column operations to guarantee that $B < A$ and $B < C$. The ring $\mathbb{K}_n[t^{\pm 1}]$ has the Euclidean algorithm. We compute the quotients q_1, \dots, q_{n+2} and remainders $r_1 \dots r_{n+1}$:

$$\begin{aligned} c &= q_1 b + r_1 \\ b &= q_2 r_1 + r_2 \\ r_1 &= q_3 r_2 + r_3 \\ &\vdots \\ r_{n-1} &= q_{n+1} r_n + r_{n+1} \\ r_n &= q_{n+2} r_{n+1} + 0 \end{aligned}$$

Let O_i for $i = 1, \dots, n+2$ denote the row operation which adds $-q_i$ times row $\frac{3}{2} + \frac{1}{2}(-1)^i$ to the other row. The first operation is

$$\begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \xrightarrow{O_1} \begin{pmatrix} a & b \\ -q_1 a & r_1 \end{pmatrix}$$

Note that O_1 takes P out of the set of upper-triangular matrices, but the full sequence of the O_i almosts takes P back to upper-triangular. To see this, define a recursion relation by $f_{-1} = 0$, $f_0 = a$, and $f_k = f_{k-2} - q_k f_{k-1}$. We see immediately that the degrees satisfy $F_i = Q_i + F_{i-1}$ for $i \geq 1$, since $Q_i > 0$, and we see that

$$\begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \xrightarrow{O_1} \begin{pmatrix} f_0 & b \\ f_1 & r_1 \end{pmatrix} \xrightarrow{O_2} \begin{pmatrix} f_2 & r_2 \\ f_1 & r_1 \end{pmatrix} \xrightarrow{O_3} \begin{pmatrix} f_2 & r_2 \\ f_3 & r_3 \end{pmatrix} \xrightarrow{O_4} \begin{pmatrix} f_4 & r_4 \\ f_3 & r_3 \end{pmatrix} \dots$$

Since $r_{n+2} = 0$, this sequence of operations ends at either

$$\begin{pmatrix} f_{n+1} & r_{n+1} \\ f_{n+2} & 0 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} f_{n+2} & 0 \\ f_{n+1} & r_{n+1} \end{pmatrix}$$

depending on the parity of n . Let S denote the operation which either swaps the two columns or swaps the columns and swaps the rows, so that

$$S(O_{n+2} \dots O_1 P) = \begin{pmatrix} r_{n+1} & f_{n+1} \\ 0 & f_{n+2} \end{pmatrix}$$

is upper-triangular. Perform a column operation to arrive at

$$P' = \begin{pmatrix} r_{n+1} & f'_{n+1} \\ 0 & f_{n+2} \end{pmatrix}$$

where $F'_{n+1} < R_{n+1} < B$. Note that $R_{n+1} + F_{n+2} = R_{n+1} + Q_{n+2} + F_{n+1} = R_n + F_{n+1}$ by the definitions of the r_i , q_i , and f_i . We apply this observation repeatedly to conclude that $R_{n+1} + F_{n+2} = R_1 + F_2$, but this is in turn $R_1 + Q_2 + Q_1 A = B + Q_1 + A = C + A$, which is the sum of the diagonal degrees of P .

Thus, we have described a procedure that takes $P = \begin{pmatrix} a & b \\ 0 & c \end{pmatrix}$ to $P' = \begin{pmatrix} a' & b' \\ 0 & c' \end{pmatrix}$ such that $A + C = A' + C'$ and $B > B'$. After repeatedly applying this procedure, we end at a diagonal matrix ($b' = 0$) which presents the same module as P and has the same diagonal degree sum. \square

4. THE WORD PROBLEM IN FINITELY PRESENTED $\mathbb{Z}[\mathbb{Z}]$ -MODULES

Recall that \mathbb{K}_1 is the (commutative) quotient field of the ring $\mathbb{Z}[G^{(1)}/G^{(2)}]$. In doing the $\mathbb{K}_1[t^{\pm 1}]$ -row operations to put the metabelian Fox matrix in upper-triangular form, it will be necessary to compute (left) polynomial quotients in the ring $\mathbb{K}_1[t^{\pm 1}]$. Thus, one needs to invert elements of \mathbb{K}_1 and in particular, needs to know when an element in $\mathbb{Z}[G^{(1)}/G^{(2)}]$ is zero so as not to divide by zero in \mathbb{K}_1 ! This is the difficulty of step 4 of Algorithm 1.

Let H be any group, and consider its group ring $\mathbb{Z}H$. Given an element $p = n_1 h_1 + \cdots + n_k h_k$ of $\mathbb{Z}H$, one may determine whether or not $p = 0$ by the algorithm: for $2 \leq i \leq k$, if $h_i = h_j$ for some $j < i$, combine the i th and j th terms by adding $n_j + n_i$. After running this $O(k^2)$ algorithm, $p = 0$ if and only if all remaining $n_i = 0$.

Suppose we have a presentation of the group H with generators x_i and relators r_j (neither indexing set must be finite), and let h be an arbitrary element of H , written as a word in the generators. The problem of deciding whether h is equal to the identity element (represented by the empty word) is known as the *word problem* for this presentation of H . We say this presentation of H has *solvable word problem* if there is an algorithm taking any element $h = h(x_i)$ of H and deciding whether $h = 1$ or not. We refer the reader to Stillwell's enlightening survey on the word problem [Sti82]. While much effort has been made to understand the word problem for finitely presented groups, the case of infinitely presented groups has received less attention.

We are interested in the group $H = G^{(1)}/G^{(2)}$, where $G = \pi_1(S^3 \setminus K)$. In this case H is an abelian group, and H is finitely generated if and only if the (classical) Alexander polynomial $\Delta_K(t)$ is monic. The word problem for finitely generated abelian groups is solvable by the structure theorem for finitely generated \mathbb{Z} -modules, but sadly most knots do not have monic Alexander polynomial. Recall that H has the structure of a $\mathbb{Z}[\mathbb{Z}]$ -module. Even though $\mathbb{Z}[\mathbb{Z}]$ is not a PID (and so its modules have no a priori structure theorem), the word problem for such modules is solvable.

Theorem 4. *Let H be an abelian group. Suppose that H has finite presentation P as a module over $\mathbb{Z}[\mathbb{Z}]$. Then there exists a group presentation P' for H that has solvable word problem.*

Note that P is not a presentation of H as a group, though a group presentation can be obtained from P . For each $\mathbb{Z}[\mathbb{Z}]$ -module generator x , one considers the \mathbb{Z} -many generators $t^i * x$. Similarly, each $\mathbb{Z}[\mathbb{Z}]$ -module relation gives rise to \mathbb{Z} -many group relations. These generators and relations give a group presentation P' for H .

The author could not find this result in the literature and expects it to be ‘folklore.’ The special case when H is the classical Alexander module of a knot in S^3 is well-known to knot theorists; see Sections 4.1 and 4.2. We present a full proof in Section 4.3. The proof for the general case leads to a more effective implementation of Algorithm 1 than the proofs in Sections 4.1 and 4.2.

Corollary 1. *Given a finite presentation P for a $\mathbb{Z}[\mathbb{Z}]$ -module H , the word problem in $\mathbb{Z}H$ is solvable.*

Proof. Given $p = n_1 h_1 + \cdots + n_k h_k$. One can determine whether $h_i = h_j$ since the word problem for H is solvable, and by the argument above, one can collapse p until it is no longer collapsible. \square

We would like to remark that the $O(k^2)$ algorithm to collapse $p = n_1 h_1 + \cdots + n_k h_k$ is far from optimal. In putting the metabelian Fox matrix in upper-triangular form, many row operations may be required. This may well involve inverting many elements in \mathbb{K}_1 , and each time an inversion is done, we

must check that we are not dividing by zero. The expressions $p = n_1 h_1 + \dots + n_k h_k$ get longer as more row operations are performed. An $O(k^2)$ operation done to many elements p of increasing length is very time consuming. Thus, a *normal form* for elements of H is preferred. A normal form for elements in a group is a canonical expression which allows for a quick decision of whether $h = 1$. For example, consider the polynomial multiplication $(1 + t + t^2) * (1 - t - t^2)$, and imagine for a moment that the terms (t^i) are unrelated. Multiplying the polynomials using distributivity yields $1 * 1 + -1 * t - 1 * t^2 + \dots$. One could then collapse the terms using the $O(k^2)$ algorithm described above. Now, remember that $t^1 * t^1 = t^2$ (i.e. t^i is a normal form for a monomial). Then one may quickly multiply $(1 + t + t^2) * (1 - t - t^2)$ by grouping the terms in the product by degree, i.e. using the normal form: from the distributed product $1 - t - t^2 + \dots$, one may rewrite the product (using a bit of memory) by scanning through the product only once. Here is the full product $1 - t - t^2 + t - t^2 - t^3 + t^2 - t^3 - t^4 = (1)1 + (-1+1)t + (-1-1+1)t^2 + (-1-1)t^3 + (-1)t^4$. Assuming that integer addition is instantaneous, this method for collapsing polynomials is $O(k)$, where k is the length of the polynomial expression.

4.1. The rational Alexander module. The classical integral Alexander module $\mathcal{A}_0^{\mathbb{Z}}(K)$ imbeds into the classical rational Alexander module $\mathcal{A}_0(K)$ by the map

$$\mathcal{A}_0^{\mathbb{Z}}(K) = G^{(1)}/G^{(2)} \hookrightarrow G^{(1)}/G^{(2)} \otimes_{\mathbb{Z}} \mathbb{Q} = \mathcal{A}_0(K)$$

which takes $a \mapsto a \otimes 1$. This map is an imbedding since $G^{(1)}/G^{(2)}$ is \mathbb{Z} -torsion free [Cro63, Theorem 1.3]. The rational Alexander module $\mathcal{A}_0(K)$ is a module over $\mathbb{Q}[t^{\pm 1}]$, a PID, and so $\mathcal{A}_0(K) \cong \bigoplus_{i=1}^n \mathbb{Q}[t^{\pm 1}] / \langle p_i(t) \rangle$ where $p_i(t) | p_{i+1}(t)$. Let $d = \sum_{i=1}^n \deg(p_i(t))$. One can easily construct a $\mathbb{Q}[t^{\pm 1}]$ -module isomorphism

$$\bigoplus_{i=1}^n \mathbb{Q}[t^{\pm 1}] / \langle p_i(t) \rangle \cong \mathbb{Q}^d$$

by recording the polynomial coefficients in the factors. (There is an arithmetic formula for t -action on \mathbb{Q}^d which can be understood by the reducing of polynomials modulo the $p_i(t)$ on the left-hand side).

To sum up, there is an embedding of $\mathbb{Z}[t^{\pm 1}]$ -modules $G^{(1)}/G^{(2)} \hookrightarrow \mathbb{Q}^d$, which induces an imbedding of rings

$$\mathbb{Z}[G^{(1)}/G^{(2)}] \hookrightarrow \mathbb{Z}[\mathbb{Q}^d]$$

The set $\mathbb{Z}[\mathbb{Q}^d]$ is easy to work with. Its elements are functions from \mathbb{Q}^d to the integers (with finite support), which can be implemented on a computer using dictionaries keyed by d -dimensional rational vectors and with values in the integers.

Since recognizing $\vec{0} \in \mathbb{Q}^d$ is trivial, this method of tensoring $H = G^{(1)}/G^{(2)}$ with \mathbb{Q} gives a proof of Theorem 4 in the case G is the fundamental group of a knot complement.

From a computer programming point of view, this approach is untenable, insofar as computing $\delta_1(K)$ is concerned. Encoding $\mathbb{Z}[G^{(1)}/G^{(2)}]$ as a submodule of $\mathbb{Z}[\mathbb{Q}^d]$ is quite simple, but then arithmetic operations in \mathbb{Q}^d are required to upper-triangularize the metabelian Fox matrix. To avoid numerical round-off errors, one must use arbitrary precision arithmetic in \mathbb{Q}^d , which has substantial overhead. Arbitrary precision arithmetic uses rational numbers with arbitrarily large numerators and denominators. When a modern-day computer does an arithmetic operation an integer with greater than 20 digits ($20 \approx \log(2^{64})$), it splits the integer into shorter pieces, which uses memory and takes time.

We attempted implementing this ‘tensoring with \mathbb{Q} ’ approach in step 4 of Algorithm 1, and our program exceeded the computer’s 8GB of RAM during computations for δ_1 of several nine crossing knots. Perhaps one explanation is that even relatively few additions in \mathbb{Q} can complicate the numerators and denominators: $\frac{1}{99} - \frac{1}{100} + \frac{1}{101} = \frac{10001}{999900}$. The t -action also complicates the rational numbers (cf. Example 5).

4.2. The Silver-Williams structure theorem. Given two maps $f : U \rightarrow A$, $g : U \rightarrow B$ in the category of abelian groups, one can form the *amalgamated sum*

$$A \oplus_U B := A \oplus B / \{(f(u), 0) = (0, g(u)) \text{ for all } u \in U\}$$

The amalgamated sum is a pushout in the category of abelian groups. If both f and g are injective, then A and B imbed into $A \oplus_U B$. If one considers the case $A = B$, then one can iterate the operation. If $A = B$ and f and g are injective, then this iteration is associative, i.e.

$$(B \oplus_U B) \oplus_U B \cong B \oplus_U (B \oplus_U B)$$

One can form infinite amalgamated sums as well: $\cdots \oplus_U B \oplus_U B \oplus_U \cdots$.

Given a finitely presented $\mathbb{Z}[\mathbb{Z}]$ -module H which is \mathbb{Z} -torsion free, Silver and Williams [SW12] construct explicit finitely generated abelian groups U and B and explicit maps $f, g : U \rightarrow B$. By replacing U with $U/(\ker f + \ker g)$ and B with $B/(g(\ker f) + f(\ker g))$ (and f and g by the induced maps) repeatedly until both f and g are injective. This procedure terminates after finitely many steps by the Noetherian property of finitely generated abelian groups. Silver and Williams construct a $\mathbb{Z}[\mathbb{Z}]$ -module isomorphism $H \rightarrow \cdots \oplus_U B \oplus_U B \oplus_U \cdots$. The t -action on the amalgamated sum is given by shifting the B factor to the right.

Using this isomorphism, we briefly describe an algorithm to decide whether a given element of H is zero. Given $x \in H$, apply the Silver-Williams isomorphism. The image lies in a finite range of summands $C := B \oplus_U \cdots \oplus_U B$. Since the amalgamating maps are injective, $C \hookrightarrow \cdots \oplus_U B \oplus_U B \oplus_U \cdots$. Deciding whether an element of $B \oplus_U \cdots \oplus_U B$ is zero is relatively simple. Let $A = B \oplus_U \cdots \oplus_U B$, which has one fewer B factor than C , so that $C \cong A \oplus_U B$. Then an element $y \in C$ is zero if and only if it can be written as a sum of elements of the form $(f(u), -g(u))$, for $u \in U$. In other words $y \in C$ is zero if and only if the B -component can be pulled back by g and pushed into A via f so that the sum of the A -component with this ‘swept back’ B -component is zero in A . One applies this sweeping (left) procedure until $A = B$, a finitely generated abelian group, where the detection of 0 is simple. We summarize this in the following proposition, which with the above remarks gives a solution to the word problem for the Silver-Williams group presentation of H .

Lemma 2. *An element y of $B \oplus_U \cdots \oplus_U B$ is zero if and only if y can be ‘swept’ to an element y' in the left-most B -factor and this $y' = 0$.*

While this approach allows one to detect 0 in the ring $\mathbb{Z}[G^{(1)}/G^{(2)}]$, it does not give a normal form for elements of $G^{(1)}/G^{(2)}$. For example, an element in C could be swept to the left out of C – there is no canonical B -factor to stop the sweeping. Thus, it does not speed up the $O(k^2)$ algorithm for collapsing elements of $\mathbb{Z}[G^{(1)}/G^{(2)}]$, and so we chose not to implement the Silver-Williams method to complete step 4 of Algorithm 1.

4.3. Gröbner bases. Gröbner bases over Laurent polynomial rings have been used for algorithmic computations of the elementary Alexander ideals [GVHHUE06] but not, to the author’s knowledge, to compute the entire Alexander module.

Let R denote the polynomial ring $\mathbb{Z}[r, s]$. We will use the *degree reverse lexicographic* order on monomials in R : $r^a s^b < r^c s^d$ if and only $a + b < c + d$ or $\{a + b = c + d \text{ and } d < b\}$. For example, $s < r < s^3 < rs^2 < r^2 s < r^3$. This is a total ordering on the monomials in R so that any polynomial may be written in the unique way so that the monomials are increasing. A *Gröbner basis* B for an ideal I in R is a finite, list of polynomials f_1, \dots, f_n so that division of any polynomial R by the elements of B gives a unique remainder. Here, division of f by B means, roughly, to reduce f modulo the elements of B until it can be reduced no further. More specifically, division of f by B is achieved by running through this loop: while there is an $f_i \in B$ such that $f = q \cdot f_i + g$ for some $q \neq 0$ and $\deg(g) < \deg(f)$, set $f = g$ and repeat until $q = 0$ for all choices of i . The computations of the quotients q of course depends on the choice of f_i for each pass through the loop, but the output of the loop, i.e. the remainder of f divided by B , is unique since B is a Gröbner basis. We denote $f \% I$ the reduction of $f \in R$ modulo the Gröbner basis B . It follows that $f \% I = 0$ if and only if $f \in I$.

One can construct an ordering on R^d from a monomial ordering in R and a ‘vector ordering.’ Let \vec{v} and \vec{w} be vectors in R^d , and let i (respectively, j) denote the index of the last nonzero entry of \vec{v} (respectively, \vec{w}). We say $\vec{v} < \vec{w}$ if either $i < j$ or $\{i = j \text{ and } \vec{v}(i) < \vec{w}(i)\}$. This gives a total ordering on the elements of R^d . Recall that the division algorithm in R computes quotients by looking

at the monomials in descending order. This division algorithm can be extended to R^d by defining a monomial in R^d to be an R -monomial times a standard basis vector in R^d .

The notion of Gröbner bases have been extended to finitely presented R -modules. We refer the reader to [AL94, Chapter 3] but discuss the salient points here. Given a finite presentation matrix P for an R -module M , one obtains an isomorphism $M \cong \text{coker } P$. Thus, the elements of M may be thought of as vectors in R^d modulo the column space of P . A Gröbner basis for the module M (with respect to the ordering on R^d) is a list B of vectors $\vec{v}_1, \dots, \vec{v}_n$ such that the reduction of any \vec{v} modulo B produces a unique remainder. We denote the reduction of \vec{v} modulo the Gröbner basis B by $\vec{v} \% B$. Such a basis is useful because it gives a normal form for elements of M , i.e. $\vec{v} = \vec{w}$ as elements of M if and only if $\vec{v} \% B = \vec{w} \% B$ in R^d . In particular, a Gröbner basis affords one an easy method to detect $0 \in M$, and such a basis is algorithmically computable.

We are interested not in the ring R but in its quotient $\mathbb{Z}[r, s]/\langle rs - 1 \rangle \cong \mathbb{Z}[t^{\pm 1}]$. Given a presentation matrix P for a $\mathbb{Z}[t^{\pm 1}]$ -module M , change all occurrences of t to r and t^{-1} to s . Then augment P by the matrix $(rs - 1)I$, and call the result P' . Reducing a vector \vec{v} modulo a Gröbner basis B' for the module presented by P' effectively reduces \vec{v} by the columns of P and sets $s = r^{-1}$. In other words, for any $\vec{v} \in R^d$, $\vec{v} \% B'$ is a normal form for the image of \vec{v} in M . We summarize this discussion in the following lemma.

Lemma 3. *Let H be a finitely presented $\mathbb{Z}[t^{\pm 1}]$ -module. Then there exists a Gröbner basis B for H that provides a normal form $x \% B \in \mathbb{Z}[t^{\pm 1}]^d$ for an element $x \in H$. Furthermore, $x \% B = \vec{0}$ if and only if $x = 0 \in H$.*

While polynomial division takes CPU time, this Gröbner basis method has provided a more effective implementation of Algorithm 1's step 4 than the 'tensoring with \mathbb{Q} ' method. While this method is slightly slower for the trefoil and figure-eight, it uses far less RAM and is able to calculate δ_1 for the knots 11_{n67} , 12_{n293} , and the (at most) 23 crossing knot from Example 4. We will illustrate the overhead benefits of the Gröbner basis method.

Example 5. Let P be the matrix

$$P = \begin{pmatrix} 3 - 3t & 2 - t \\ 1 - 2t & 3 - 3t \end{pmatrix}$$

as in Example 4. Let us view $P : \mathbb{Z}[\mathbb{Z}]^2 \rightarrow \mathbb{Z}[\mathbb{Z}]^2$ by $\vec{x} \mapsto \vec{x}P$. So the module P presents is $\mathbb{Z}[\mathbb{Z}]^2$ modulo the row space of P . This agrees with the convention for $\mathcal{A}_0^{\mathbb{Z}}(K)$ as established in Example 3. We aim to find the normal form of the vectors $\vec{x} = (3t^3 + 2t^2 + 2)e_1 + (5t^2 - t^3)e_2$ and $t * \vec{x}$ using the rational Alexander module and Gröbner basis methods.

Working over $\mathbb{Q}[\mathbb{Z}]$, we calculate the Smith normal form of P to be $D = LPR$, where

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 7t^2 - 13t + 7 \end{pmatrix} \text{ and } R = \begin{pmatrix} 1 & -7/3t + 5/3 \\ 0 & 1 \end{pmatrix}$$

For this example, L is irrelevant but may be computed using inverses. As a $\mathbb{Q}[\mathbb{Z}]$ -module, $M \otimes \mathbb{Q}$ is $\mathbb{Q}[\mathbb{Z}]/\langle 7t^2 - 13t + 7 \rangle$, though recall that M merely injects into $M \otimes \mathbb{Q}$ as a $\mathbb{Z}[\mathbb{Z}]$ -module. This quotient ring is isomorphic to \mathbb{Q}^2 by recording the coefficients of the linear and constant terms of a polynomial's reduction modulo $7t^2 - 13t + 7$.

Now given any element $y \in \mathbb{Z}[\mathbb{Z}]^2$ which projects to an element $\bar{y} \in M$, we have that $\phi(\bar{y}) = \bar{y}R$, where ϕ is the map that takes M to $\mathbb{Q}[\mathbb{Z}]/\langle 7t^2 - 13t + 7 \rangle$ and $\bar{}$ denotes taking the quotient modulo P or D . After composing with the isomorphism to \mathbb{Q}^2 , we see that our $\vec{x} \mapsto (-\frac{1420}{147}, \frac{281}{21})$ and $t\vec{x} \mapsto (-\frac{4691}{1029}, \frac{1420}{147})$. Notice how the t -action complicates the rational entries. Addition in this module complicates these rational numbers even more.

Let $r = t$ and $s = t^{-1}$. Using Macaulay2 [GS], we compute a Gröbner basis as described above to be

$$G = \left\{ \begin{pmatrix} 7r + 7s - 13 \\ 0 \end{pmatrix}, \begin{pmatrix} 7s^2 - 13s + 7 \\ 0 \end{pmatrix}, \begin{pmatrix} rs - 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 7s - 5 \\ 3 \end{pmatrix}, \begin{pmatrix} 5s - 4 \\ s + 1 \end{pmatrix}, \begin{pmatrix} 3r + 7s - 8 \\ r + 1 \end{pmatrix} \right\}$$

Macaulay2 computes $\vec{x} \% G = (-r^3 + 3r^2 - 2r - 14s + 14, 0)$ and $(r\vec{x}) \% G = (-r^4 + 3r^3 - 2r^2 - 14s + 12, 0)$.

In the $\otimes \mathbb{Q}$ method, the addition and t -action have inherent definitions in \mathbb{Q}^2 . This is easy to implement and feels like it should be fast. However, the rational numbers become complicated after a few operations. Each subsequent operation takes increasingly more time, as evidenced by Example 6. In practice, this method's implementation exhausts our machine's memory during even simple δ_1 calculations.

In the Gröbner basis method, the addition and t -action do not have inherent definitions, i.e. $(\vec{x} + \vec{y})\%G \neq \vec{x}\%G + \vec{y}\%G$ and $(t\vec{x})\%G \neq t(\vec{x}\%G)$. After every addition and t -action, one must divide the result by the Gröbner basis, which takes CPU time. However, the form $\vec{x}\%G$ is simpler – requiring less RAM than its \mathbb{Q}^2 counterpart – and we find the Gröbner basis implementation of encoding the ring $\mathbb{Z}[G^{(1)}/G^{(2)}]$ more effective.

Example 6. The t -action on \mathbb{Q}^2 in the above example is given by $t * (a, b) = (b + \frac{13}{7}a, -a)$. Consider the elements $v_0 = (-1420/147, 281/21)$ and $x = (1282/147, -167/21)$ of \mathbb{Q}^2 . These are the images under the isomorphism $\mathbb{Z}[\mathbb{Z}]^2 / \text{row } P \rightarrow \mathbb{Q}^2$ of the elements $(3t^3 + 2t^2 + 2, 5t^2 - t^3)$ and $(t^2 - 3t + 7, 4t^2 + 5t^3 - 2)$, respectively. Define a sequence v_i by $v_n = t * v_{n-1} + x$.

In the algorithm to compute δ_1 , one must perform row operations on a matrix whose entries lie in $\mathbb{K}_1[t^{\pm 1}]$. The coefficient of a monomial of one entry of this matrix is a formal quotient of elements of $\mathbb{Z}[G^{(1)}/G^{(2)}]$. The numerator and denominator of this quotient may be quite long. Each term in the numerator is an integer times an element of $G^{(1)}/G^{(2)}$. If we think of $G^{(1)}/G^{(2)}$ as a subset of \mathbb{Q}^2 , it is tempting to absorb this integer into the vector, but this is not permissible in the group ring $\mathbb{Z}[\mathbb{Q}^2]$. A single row operation on this matrix necessitates adding many monomials and hence many coefficients in \mathbb{K}_1 . Each addition $a/b + c/d = (ad + bc)/bd$ in \mathbb{K}_1 involves three multiplications in the group ring $\mathbb{Z}[\mathbb{Q}^2]$. Each group ring multiplication involves *many* group operations in \mathbb{Q}^2 , depending on the number of terms in the $\mathbb{Z}[\mathbb{Q}^2]$ -elements which are multiplied. Note that each row operation on the matrix vastly increases the number of group operations in \mathbb{Q}^2 that must be done. After a few row operations, there are a multitude of $G^{(1)}/G^{(2)}$ elements involved. If we think of these as elements of \mathbb{Q}^2 , the numerators and denominators of the entries of each vector in \mathbb{Q}^2 grow longer as the number of \mathbb{Q}^2 operations increases. This, we believe, is why the ‘tensoring with \mathbb{Q} ’ implementation of step 4 exhausts our computer of all available memory.

The sequence v_n is intended to indicate how complicated the rational numbers become after many arithmetic operations. We do not foresee the v_n arising in a δ_1 computation. While the rational model for $G^{(1)}/G^{(2)}$ offers a faster implementation for the $\mathbb{Z}[G^{(1)}/G^{(2)}]$ -operations than the Gröbner basis model – at least when few operations are required – the rational model slows down significantly when many operations are performed. We computed v_n to $n = 50000$ and noticed a significant slowing down of the computation as n increased. See the table in Figure 6. The CPU time required to compute v_n appears to be exponential in n .

Note that the terms of this sequence v_n written in the Gröbner basis model will also become complicated as n increases. By the Gröbner basis in Example 5, the second entry in v_n will be an integer, and the first entry will be a polynomial in r (with perhaps one or two s terms) whose degree grows sub-linearly in n and most (all but two) of whose coefficients have absolute value less than $7/2$ (Macaulay2 computes a quotient so that the remainder's leading coefficient is as small in absolute value as possible, i.e. $13r \% 5r = -2r$). Note that the second entry has absolute value less than $3/2$. Thus, v_n can be stored in a computer by approximately n digits. This is more efficient than the rational method, at least experimentally, for approximately n digits were used *just for the denominator of one entry* in the \mathbb{Q}^2 representation of v_n , according to the table above.

5. COMPUTATIONS

Using a 2.5 GhZ Intel i5 processor with four cores and 8 GB of RAM, we made the following δ_1 calculations. The implementation is discussed in Section 6. We obtained the PD code for all knots using KnotInfo [CL13]. For the 23 crossing Example 14, we used the program Link Sketcher, which we found on KnotInfo, to compute the PD Code.

Step i	CPU time to compute v_{1000i} , in seconds	Digits in denominator of first entry of v_{1000i}
1	0.048321	844
2	0.146888	1689
3	0.315882	2534
4	0.57017	3380
5	0.921086	4225
10	4.870255	8450
20	28.885591	16901
30	84.075005	25352
40	181.961961	33803
45	249.442993	38029
46	265.189638	38874
47	281.562427	39719
48	298.729361	40564
49	316.309955	41409
50	334.467209	42254

Example 7. The trefoil knot 3_1 has PD code $X[1, 5, 2, 4], X[3, 1, 4, 6], X[5, 3, 6, 2]$. From this PD code, our program calculated $\delta_1(3_1) = 1$ in 0.2128 seconds.

Example 8. The figure-eight knot 4_1 has PD code $X[4, 2, 5, 1], X[8, 6, 1, 5], X[6, 3, 7, 4], X[2, 7, 3, 8]$. Our program calculated $\delta_1(4_1) = 1$ in 0.5616 seconds.

Example 9. The 2, 5-torus knot 5_1 has PD code $X[2, 8, 3, 7], X[4, 10, 5, 9], X[6, 2, 7, 1], X[8, 4, 9, 3], X[10, 6, 1, 5]$. Our program calculated $\delta_1(5_1) = 3$ in 9.5816 seconds.

Example 10. The knot 5_2 has PD code $X[1, 5, 2, 4], X[3, 9, 4, 8], X[5, 1, 6, 10], X[7, 3, 8, 2], X[9, 7, 10, 6]$. Our program calculated $\delta_1(5_2) = 1$ in 1.4597 seconds.

Example 11. The stevedore knot 6_1 has PD code $X[1, 7, 2, 6], X[3, 10, 4, 11], X[5, 3, 6, 2], X[7, 1, 8, 12], X[9, 4, 10, 5], X[11, 9, 12, 8]$. Our program calculated $\delta_1(6_1) = 1$ in 2.7247 seconds.

Example 12. The knot 11_{n67} has PD code $X[4, 2, 5, 1], X[8, 4, 9, 3], X[11, 17, 12, 16], X[14, 5, 15, 6], X[6, 15, 7, 16], X[9, 19, 10, 18], X[17, 11, 18, 10], X[19, 1, 20, 22], X[13, 20, 14, 21], X[21, 12, 22, 13], X[2, 8, 3, 7]$. This PD code produces a Wirtinger presentation with 11 generators, and our program would attempt to put a 10×10 matrix over $\mathbb{K}_1[t^{\pm 1}]$ in upper-triangular form. However, 11_{n67} is a 3-bridge knot, and thus its fundamental group has a presentation with 3 generators and 2 relations. Using Tietze transformations in GAP [GAP12], we were able to reduce the Wirtinger presentation to the generators x_1, x_2, x_3 and relations $x_1^{-1}x_2x_3^{-1}x_2^{-2}x_1x_3x_2x_1^{-1}x_2x_3^{-1}x_2^{-2}x_1x_2^{-1}x_3^{-1}x_1^{-1}x_2^2x_3x_2^{-1}x_1x_2x_3x_2^{-1}$ and $x_1x_3^{-1}x_1^{-1}x_2x_3^{-1}x_2^{-2}x_3^{-1}x_1^{-1}x_2^2x_3x_2^{-1}x_1^2x_3x_1^{-1}x_2x_3^{-1}x_2^{-2}x_1x_3x_2^2x_3x_2^{-1}$. In this presentation, each x_i maps to a generator of \mathbb{Z} , and so a presentation with a ‘nice’ (as in Example 3) generating set can easily be obtained. This results in a 2×2 metabelian Fox matrix that must be put in upper triangular form. Of course, the entries in the 2×2 matrix are more complicated than in the original 10×10 matrix, but the δ_1 computation is faster.

Given the reduced Wirtinger presentation, our program calculated $\delta_1(11_{n67}) = 3$ in 11324 seconds. As mentioned around Theorem 2, this is the first known knot in the tables with $\delta_1 > \delta_0$.

Example 13. The knot 12_{n293} has PD code $X[1, 4, 2, 5], X[3, 10, 4, 11], X[5, 12, 6, 13], X[16, 8, 17, 7], X[9, 2, 10, 3], X[11, 8, 12, 9], X[20, 13, 21, 14], X[6, 16, 7, 15], X[24, 17, 1, 18], X[22, 19, 23, 20], X[14, 21, 15, 22], X[18, 23, 19, 24]$. Using GAP, we found a simplified presentation for the fundamental group of the knot complement with generators a, b, c and relators $b^2ab^{-2}a^{-1}c^{-1}bacab^2a^{-1}b^{-2}a^{-1}c^{-1}a^{-1}b^{-1}cab$ and $b^{-1}a^{-1}c^{-1}bab^{-1}cabcab^{-1}a^{-1}c^{-1}bab^{-1}a^{-1}c^{-1}ba^{-1}b^{-1}caba^{-1}c^{-1}$. Here a maps to a generator under abelianization, and b and c map to zero. Our program calculated $\delta_1(12_{n293}) = 3$ in 8393 seconds. This is another example of a knot with $2 \cdot \text{genus} = 1 + \delta_1 > 2 \cdot \delta_0$.

Example 14. Let K denote the knot from Example 4. The diagram for K in Figure 1 has 23 crossings, and its PD code is $X[34, 1, 35, 2], X[2, 33, 3, 34], X[32, 3, 33, 4], X[4, 31, 5, 32], X[30, 5, 31, 6], X[6, 29, 7, 30], X[7, 40, 8, 41], X[8, 20, 9, 19], X[22, 10, 23, 9], X[37, 10, 38, 11], X[11, 36, 12, 37], X[12, 24, 13, 23], X[46, 13, 1, 14], X[14, 45, 15, 46], X[44, 15, 45, 16], X[16, 43, 17, 44], X[42, 17, 43, 18], X[18, 41, 19, 42], X[27, 20, 28, 21], X[21, 26, 22, 27], X[24, 36, 25, 35], X[38, 26, 39, 25], X[28, 40, 29, 39]$. Using Tietze transformations in GAP, we were able to reduce the Wirtinger presentation with three generators x_1, x_2, x_3 and three relations $x_1x_2^{-1}x_1x_2x_1^{-1}x_2x_3^{-1}x_2x_3^{-1}x_2^{-1}x_3x_2^{-1}$ and $x_3^{-1}x_1x_2^{-1}x_1x_2x_1^{-1}x_2x_1^{-1}x_3x_1^{-1}x_3^{-1}x_1$.

Our program calculated $\delta_1(K) = 1$ in 0.7698 seconds.

6. AN IMPLEMENTATION OF THE ALGORITHM

We implemented Algorithm 1 in Sage 5.3. Most of the classes and functions are written in Python 2.7.2, and we used Macaulay2 version 1.4 to compute the Gröbner basis for $\mathcal{A}_0^{\mathbb{Z}}(K)$. Sage incorporates Python and Macaulay2, and so our program can be executed in Sage with one click of a button. Sage [S⁺12], Python [Pyt12], and Macaulay2 [GS] are open source and cross-platform.

Our implementation can take as an input the PD code for a knot diagram or a Wirtinger presentation for the knot complement's fundamental group. We use a script written by REU students to convert the PD code to a Wirtinger presentation [REU11]. At this point, the algorithm runs quite similarly to Example 3. After switching to a convenient generating set for the fundamental group, the Fox matrix is then calculated, and a presentation matrix (the abelian Fox Matrix) for $G^{(1)}/G^{(2)} \cong \mathcal{A}_0^{\mathbb{Z}}(K)$ is recorded. The abelian Fox matrix is fed to Macaulay2, which computes a Gröbner basis for the classical Alexander module. Separately, the entries of the original Fox matrix are written modulo $G^{(2)}$, using a splitting which treats the first generator of the Wirtinger presentation as t , so $G/G^{(2)} \cong G^{(1)}/G^{(2)} \rtimes \mathbb{Z}$, as in equation 2. Our program finds a column to delete by reducing the $G^{(1)}/G^{(2)}$ -generators modulo the Gröbner basis computed earlier, and deletes it, yielding the metabelian Fox matrix, which presents $\mathcal{A}_1(K)$ as a $\mathbb{K}_1[t^{\pm 1}]$ -module. Finally, this metabelian Fox matrix is put in upper-triangular form by row and column operations, and the degrees of the polynomials are computed and added, giving $\delta_1(K)$.

We wrote classes for the elements of π_1 , $G/G^{(2)}$, $\mathbb{Z}\Gamma_1$, \mathbb{K}_1 , and $\mathbb{K}_1[t^{\pm 1}]$, and functions to add and/or multiply elements in each of those sets, which culminated in the polynomial ring operations. Throughout the upper-triangularization process, we need the degree function in $\mathbb{K}_1[t^{\pm 1}]$, which amounts to recognizing when the coefficient of a term is zero or not, hence the importance of Section 4. In $\mathbb{K}_1[t^{\pm 1}]$, we need to compute quotients of polynomials on the left and right so that we can do row and column operations to upper-triangularize the metabelian Fox matrix.

We implemented group ring elements in $\mathbb{Z}[G^{(1)}/G^{(2)}]$ as Python dictionaries. A *dictionary* is a function whose domain consists of *keys*. The range of the function is the set of *values* for the dictionary. In Python, keys must be hashable, so that dictionary lookup time (recalling the value of the function given a key) is constant time. A group ring element in $\mathbb{Z}[G^{(1)}/G^{(2)}]$ can then be thought of as a dictionary keyed by elements of $G^{(1)}/G^{(2)}$ with values in \mathbb{Z} . Multiplication of elements in $\mathbb{Z}[G^{(1)}/G^{(2)}]$ follows the rule $(ax + by)(cz + dw) = ab(x + z) + ad(x + w) + bc(y + z) + bd(y + w)$ (we use here that the group $G^{(1)}/G^{(2)}$ is abelian). Python dictionaries can be used to multiply and group like terms, since we have a normal form for elements of $G^{(1)}/G^{(2)}$. We include the pseudo-code for our group ring multiplication function:

```
def multiply_dictionaries(dict1, dict2):
    product = {}
    for v in keys of dict1:
        for w in keys of dict2:
            # v and w are vectors reduced by the Groebner basis GB;
            product_key = (v+w) % GB;
            product[product_key] = 0;
    for v in keys of dict1:
        for w in keys of dict2:
```

```

        product[(v+w) % GB] = product[(v+w) % GB] + dict1[v]*dict2[w];
    for k in keys of product:
        if product[k] == 0:
            delete product[k];
    return product

```

The first for loop creates all of the sums of group elements that occur in the product in the group ring. The second for loop computes all the integer coefficients in the product. The third for loop removes all terms in the product with coefficient zero. If l is the length of the first factor and r is the length of the second factor, this function runs through $3lr$ loops, at most. The returned product is completely collapsed. Alternatively, one could implement group ring elements as lists. Multiplication and collapsing of lists must run through at most $lr + (lr)^2$ loops.

A Sage worksheet for computing δ_1 is available at <https://pdhorn.expressions.syr.edu/software/>.

REFERENCES

- [AL94] William W. Adams and Philippe I. Loustanaun, *An introduction to Gröbner bases*, Graduate Studies in Mathematics, vol. 3, American Mathematical Society, Providence, RI, 1994.
- [CL13] Jae Choon Cha and Charles Livingston, *Knotinfo: Table of knot invariants*, <http://www.indiana.edu/~knotinfo>, April 2013.
- [Coc04] Tim D Cochran, *Noncommutative knot theory*, *Algebr. Geom. Topol.* **4** (2004), 347–398.
- [Coh85] Paul M Cohn, *Free rings and their relations*, 2 ed., London Mathematical Society Monographs, vol. 19, Academic Court [Harcourt Brace Javonovich Publishers], London, 1985.
- [COT03] Tim D Cochran, Kent E Orr, and Peter Teichner, *Knot concordance, Whitney towers and L^2 -signatures*, *Ann. of Math. (2)* **157** (2003), no. 2, 433–519.
- [Cro63] Richard H Crowell, *The group G'/G'' of a knot group G* , *Duke Math. J.* **30** (1963), no. 349–354.
- [Cul] Marc Culler, *Gridlink 2.0*, software, available at <http://www.math.uic.edu/~culler/gridlink>.
- [Fox53] Ralph H. Fox, *Free differential calculus. I. Derivation in the free group ring.*, *Annals of Mathematics* **57** (1953), no. 3, 547–560.
- [FV11] Stefan Friedl and Stefano Vidussi, *A survey of twisted alexander polynomials*, *Contributions in Mathematical and Computational Sciences*, vol. 1, pp. 45–94, Springer, Heidelberg, 2011.
- [GAP12] The GAP-Group, *GAP – Groups, Algorithms, and Programming, Version 4.5.7*, 2012, <http://www.gap-system.org/>.
- [GS] Daniel R. Grayson and Michael E. Stillman, *Macaulay2, a software system for research in algebraic geometry*, Available at <http://www.math.uiuc.edu/Macaulay2/>.
- [GVHHUE06] Jesús Gago-Vargas, Isabel Hartillo-Hermoso, and José María Ucha-Enríquez, *Algorithmic invariants for Alexander modules*, *Computer algebra in scientific computing* (Victor G. Ganzha, Ernst W. Mayr, and Evgenii V. Vorozhtsov, eds.), *Lecture Notes in Computer Science*, vol. 4194, Springer, Berlin, 2006, pp. 149–154.
- [Har05] Shelly L Harvey, *Higher-order polynomial invariants of 3-manifolds giving lower bounds for the Thurston norm*, *Topology* **44** (2005), 895–945.
- [Hol10] Erik Holum, *Calculating the degree of higher order Alexander polynomials*, Undergraduate honors thesis, Wesleyan University, Middletown, CT, 2010.
- [Lew72] Jacques Lewin, *A note on zero divisors in group-rings*, *Proc. Amer. Math. Soc.* **31** (1972), no. 2, 357–359.
- [LM08] Constance Leidy and Laurentiu Maxim, *Obstructions on fundamental groups of plane curve complements, Real and complex singularities*, *Contemporary Mathematics*, vol. 459, American Mathematical Society, Providence, RI, 2008, pp. 117–130.
- [Pyt12] The Python Software Foundation, *Python, Version 2.7.2*, 2012, <http://www.python.org/>.
- [REU11] REU Group in Noncommutative Knot Theory, *PDcodetoWirtGUI.py*, Columbia University Topology RTG, 2011, <http://pdhorn.expressions.syr.edu/>.
- [Rol76] Dale Rolfsen, *Knots and links*, Publish or Perish, Berkeley, CA, 1976.
- [S+12] William A. Stein et al., *Sage Mathematics Software, Version 5.3*, The Sage Development Team, 2012, <http://www.sagemath.org/>.
- [Sti82] John Stillwell, *The word problem and the isomorphism problem for groups*, *Bull. Amer. Math. Soc. (New Series)* **6** (1982), no. 1, 33–56.
- [Str74] Ralph Strebel, *Homological methods applied to the derived series of groups*, *Comment. Math. Helv.* **49** (1974), 302–332.
- [SW12] Daniel Silver and Susan Williams, *On modules over Laurent polynomial rings*, *Journal of Knot Theory and its Ramifications* **21** (2012), no. 1, 6 pages.

- [Wal78] Friedhelm Waldhausen, *Recent results on sufficiently large 3-manifolds*, Algebraic and geometric topology (Proc. Sympos. Pure Math., Stanford Univ., Stanford, CA, 1976), Part 2, Proceedings of Symposia in Pure Mathematics, vol. XXXII, American Mathematical Society, Providence, RI, 1978, pp. 21–38.

DEPARTMENT OF MATHEMATICS, SYRACUSE UNIVERSITY, 215 CARNEGIE BUILDING, SYRACUSE, NY 13244-1150

E-mail address: `pdhorn@syr.edu`

URL: `https://pdhorn.expressions.syr.edu/`